Dokumentacja projektu
mPlayer
Programowanie III

Dawid Bitner, Marcin Krupa IA

22 stycznia 2019

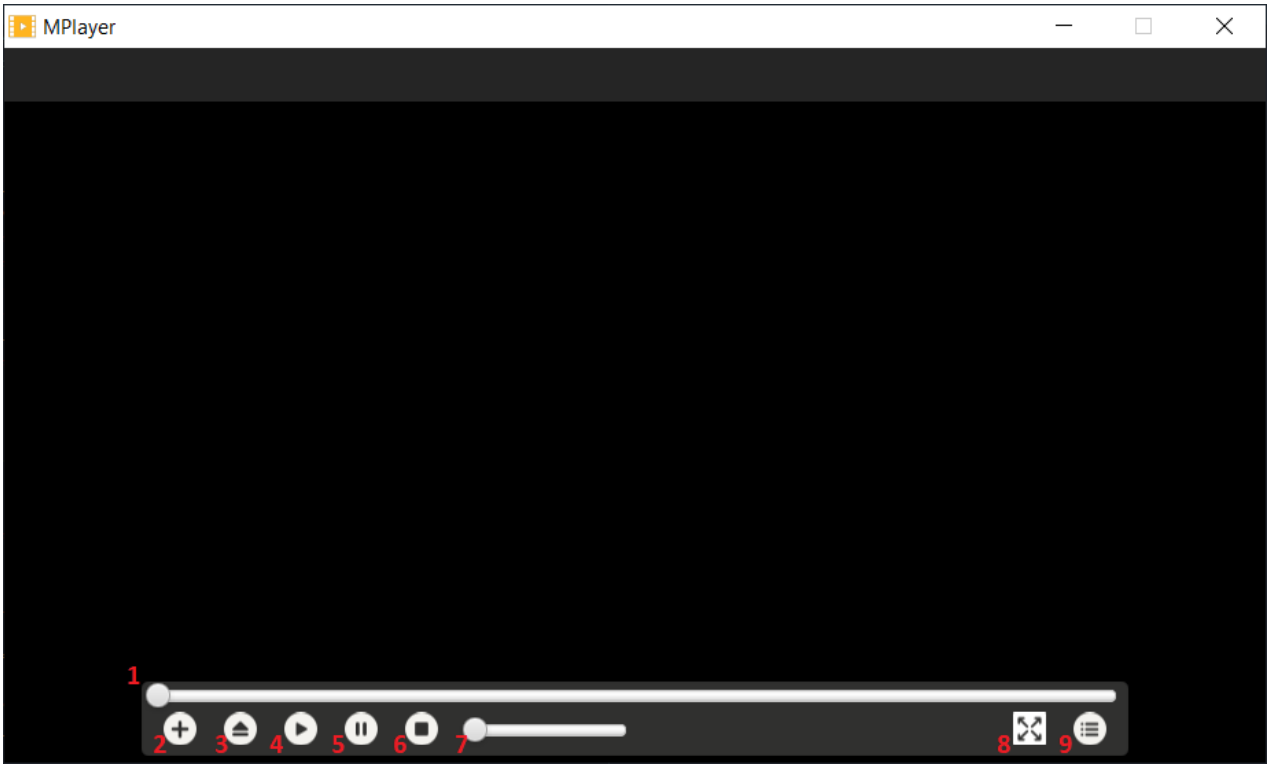
Spis treści

1 Część I

1.1 Opis programu

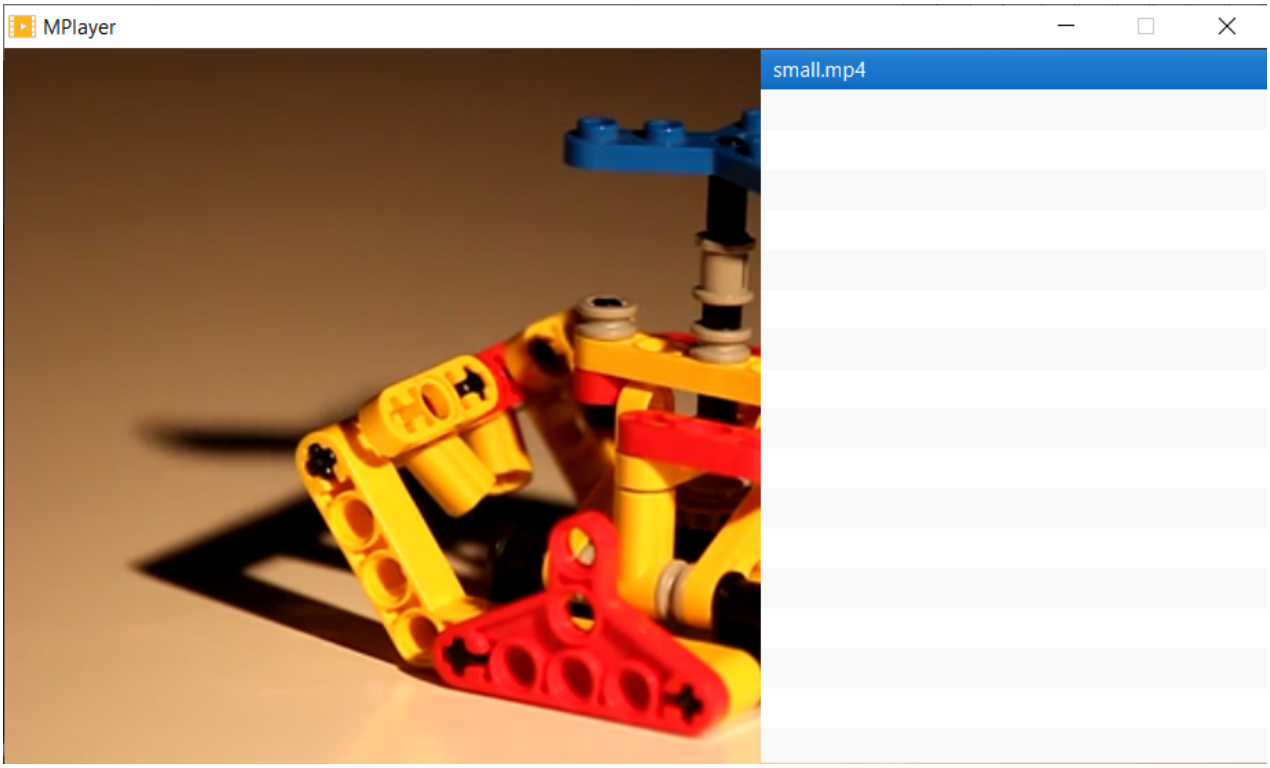
Celem programu jest umożliwienie użytkownikowi odtworzenie wybranych przez niego plików wideo zapisanych w formacie *.mp4*.

1.2 Instrukcja obsługi



[1] suwak postępu
[2] dodanie pliku do playlisty
[3] dodanie zawartości folderu do playlisty
[4] odtwarzanie
[5] pauza
[6] zatrzymanie odtwarzania
[7] suwak poziomu głośności
[8] przełączanie pomiędzy trybem pełnoekranowym
[9] lista odtwarzania

Odtwarzacz z widocznym wysuniętym menu zawierającym listę odtwarzania:



2 Część II

2.1 Część techniczna

Program uruchamiamy poprzez wywołanie pliku *mPlayer.jar*. W celu zapewnienia poprawnego działania programu potrzebujemy systemu operacyjnego z zainstalowanym środowiskiem *Java* w wersji *8* lub nowszej. Program został napisany przy pomocy zintegrowanego środowiska programistycznego *NetBeans IDE 8.2*, strona wizualna projektu została stworzona przy pomocy *JavaFX Scene Builder 8.5.0*, darmowego narzędzia do edycji plików technologii *JavaFX* firmy *Gluon*, na której to został oparty projekt.
Testy programu zostały przeprowadzone na systemach operacyjnych *Windows* 7 64-bit, *Windows 8.1* 64-bit, *Windows 10* 64-bit.

2.2 Podział projektu

2.2.1 MPlayer.java

Plik ten zawiera główną klasę programu o nazwie *MPlayer*, dziedziczącą po wbudowanej klasie *Application* z rodziny *JavaFX*. Znajdują się tutaj główne metody tj. *main()* i *start()*. Plik zawiera obiekt:

```
AnchorPane root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
```

Odpowiada on za pobranie hierarchii obiektów z pliku XML, które są obiektami graficznymi uwzględnionymi w projekcie rozszerzając jego funkcjonalność. Obiekt *root* jest obiektem klasy *AnchorPane*, jest zatem głównym panelem projektu.

```
Scene scene = new Scene(root);
```

Obiekt *root* służy do zainicjalizowania obiektu klasy *Scene*, który jest kontenerem dla całej zawartości sceny projektu, który jest wyświetlany użytkownikowi. O ile *root* zawiera bardziej strukturę logiczną projektu, o tyle obiekt *scene* ma tę strukturę wyświetlić użytkownikowi.

```
FXMLDocumentController controller = new FXMLDocumentController();
```

Kolejnym istotnym ważnym obiektem jest obiekt *controller* klasy *FXMLDocumentController*. Inicjalizuje on wszelkie metody sterujące aplikacją zawarte w pliku *FXMLDocumentController.java*.

Następnie, poprzez wywołanie odpowiednich metody klasy *Stage* dla obiektu *stage*, będącego nadrzędnym kontenerem całego projektu zostały określone takie właściwości jak startowy rozmiar okna, jego nazwa, ikona programu, czy to ja ma się zachowywać gdy zostanie wychwycone podwójne naciśnięcie lewego przyciska myszy.

2.2.2 FXMLDocument.fxml

Plik wygenerowany na podstawie zmian dokonywanych za pomocą *JavaFX Scene Builder* na strukturze obiektów XML. Plik jest oparty o *XML/FXML*. Zawiera on informacje o obiektach umieszczonych w projekcie, ich hierarchii oraz edycji obiektów w kwestiach takich jak zmiana układu, prywatnych właściwości obiektów, czy kodu (przypisanie ID czy wywołanie odpowiednich metod typu Action Methods).

2.2.3 FXMLDocumentController.java

Plik ten zawiera najważniejsze metody sterujące aplikacją. Wsród nich:

`checkResolution()`

Funkcja ta ma za zadanie określenie rozdzielczości ekranu użytkownika. Korzysta ona z klasycznego rozwiązania którym jest obiekt klasy *Dimensions*, pobierający rozdzielczość z klasy *Toolkit* z funkcji *getDefaultToolkit* oraz *getScreenSize*. Na podstawie uzyskanych danych, program jest w stanie dostować rozmiar okna do rodzielczości stosowanej przez użytkownika.

`addSingleFile()`

Funkcja dodaje jeden wybrany plik przez użytkownika do playlisty. Wykorzystuje ona klasę FileChooser, oraz jej metody jak *ExtensionFilter* do filtrowania rozszerzeń plików czy *showOpenDialog* do wyświetlenia okienka eksploratora do wyboru pliku.

`addFolder()`

Działa podobnie do funkcji *addSingleFile()*, jednak w tym przypadku korzysta z klasy DirectoryChooser. Wybrany folder, a dokładniej pliki z odpowiednim rozszerzeniem, przekazywane są do listy która jest następnie przekazywana do obiektu *playlist* klasy *ListView*, będącym menu wyboru załadowanych plików.

`selectItem()`

Funkcja pozwala wybrać interesujący użytkownika plik wideo poprzez dwuklik. Nazwa pliku jest wyświetlana także w panelu na górze aplikacji.

`fullScreen()`

Pozwala zmienić tryb na pełnoekranowy z wykorzystaniem przycisku z punktu [8] opisu obsługi programu. Tryb pełnoekranowy można także wywołać dwuklikiem.

`openListView()`

Otwiera playlistę z prawej strony aplikacji.

`cBarOnMouseEntered() oraz cBarOnMouseExited()`

Control Bar aplikacji, czyli panel odpowiedzialny za wszystkie przyciski sterujące aplikacją. Reaguje na najechanie myszką przez użytkownika.

`PaneOnMouseEntered() oraz PaneOnMouseExited()`

Panel na górze aplikacji. Reaguje na najechanie myszką przez użytkownika. Wyświetla aktualnie grany plik wideo.

`KeyPressed()`

Funkcja która odpowiada za zatrzymywanie/odtwarzanie pliku wideo po naciśnięciu spacji przez użytkownika.

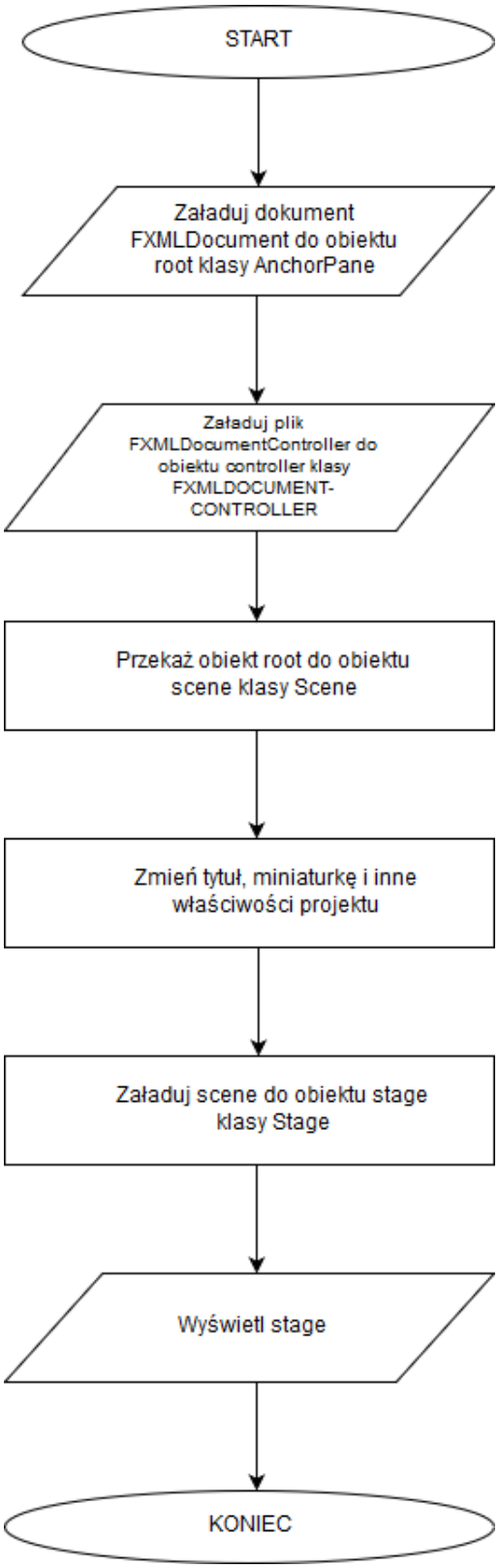
`Funkcja główna: playFile()`

Funkcja główna odpowiedzialna za odtwarzanie pliku wideo. Korzysta ona z obiektów: *media*, *mediaPlayer* oraz *mediaView* odpowiednio klas *Media*, *MediaPlayer* oraz *MediaView*. Jednym z jej zadań, poza odtwarzaniem pliku, jest kontrola wielkości okna aplikacji oraz: zmian w pasku progresu odtwarzania, aktualizacji czasu odtwarzania pliku, zmiany głośności odtwarzania poprzez wykorzystanie słuchaczy, tj. interfejsów klasy *ActionListener*. Funkcja ta także odpowiada za zmiany w progresie odtwarzania, gdy użytkownik kliknie w interesujący go fragment na pasku odtwarzania wideo przeskoczy do tego momentu.

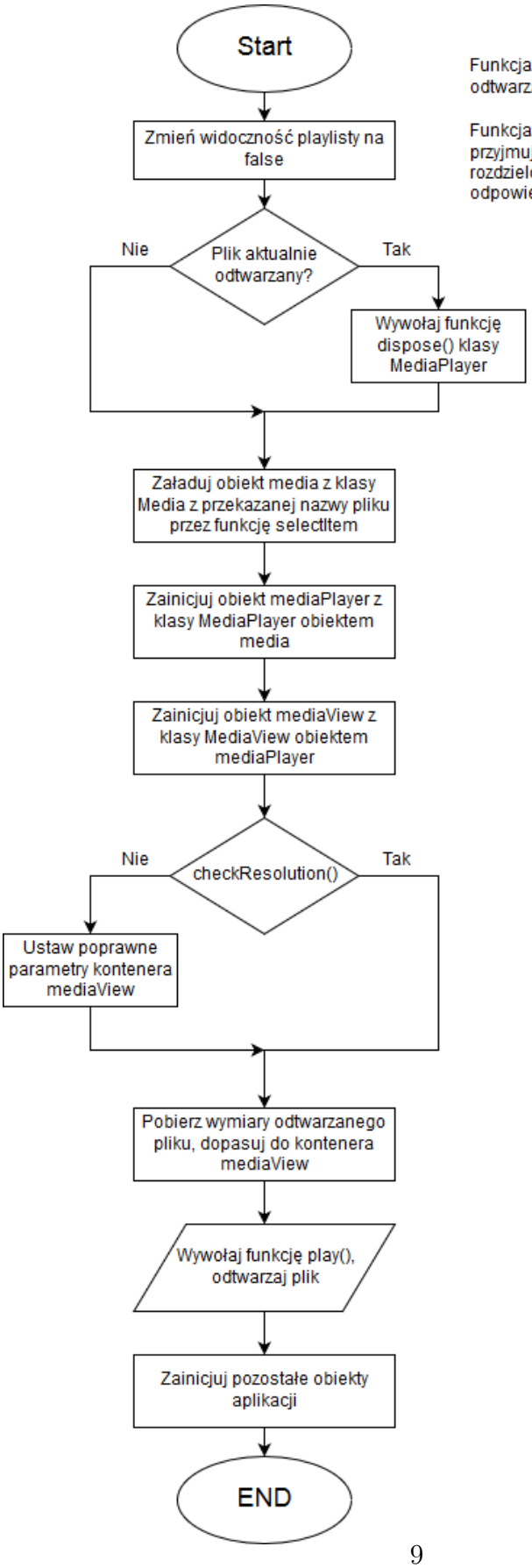
2.2.4 Kaskadowy arkusz styli: style.css

Zawiera zmiany szaty graficznej projektu. Korzysta z technologii *Cascading Style Sheets*.

2.3 Ogólny schemat blokowy projektu



2.4 Schemat funkcji playFile()



Funkcja **dispose()** usuwa z pamięci odtwarzany plik

Funkcja **checkResolution()** domyślnie przyjmuje, że użytkownik ma ustawioną rozdzielczość 16:9, ale jeśli jest inna, zwraca odpowiednią wartość

3 Zaimplementowane biblioteki

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.*;
import javafx.animation.FadeTransition;
import javafx.beans.Observable;
import javafx.beans.binding.Bindings;
import javafx.beans.property.DoubleProperty;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.fxml.*;
import javafx.scene.Parent;
import javafx.scene.control.*;
import javafx.scene.control.cell.TextFieldListCell;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.media.*;
import javafx.stage.*;
import javafx.util.Duration;
import javafx.scene.layout.*;
```